

## P6 利用 Verilog 开发 MIPS 流水线处理器 plus

### 一. 整体结构：

流水线处理器包括流水寄存器、各级组合逻辑以及各级控制器三大部分

它们均放在 mips.v 层次下，其中 code.txt 中存储相应指令码

处理器为 32 位处理器，支持的指令集为：addu,subu,ori,lw,sw,beq,lui,j,jal,  
jr,nop

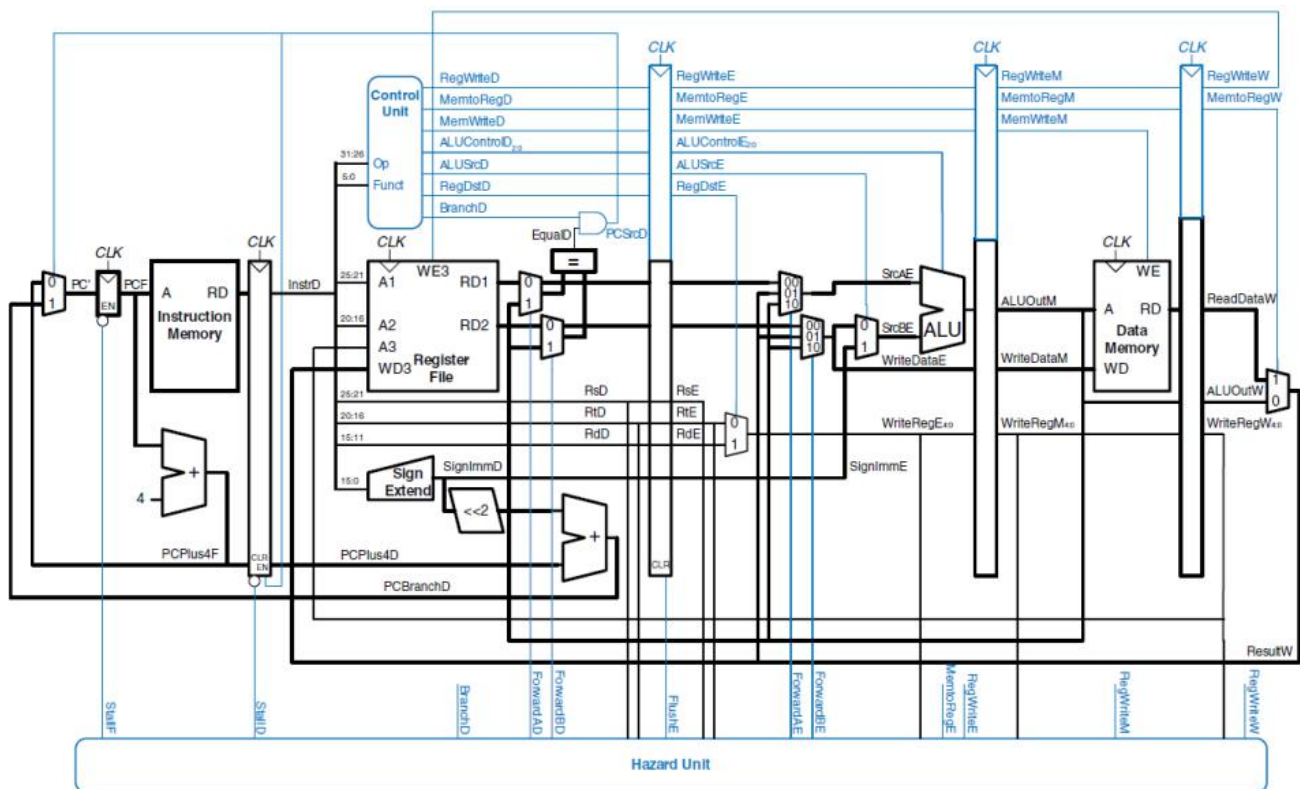


Figure 7.58 Pipelined processor with full hazard handling

二. 数据通路部分

			addu	subu	beq	lw	sw	ori	lui	j	jal	jr	MUX	控制
IF级部件	部件	输入												
	PC													
	ADD4		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC		
D级对PC更新	IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC		
	PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	PC1_MUX	PC1S+1
	IR_D		IM	IM	IM	IM	IM	IM	IM	IM	IM	IM		
F/D级流水线寄存器	PC4_D				ADD4						ADD4	ADD4		
	PCB_D													
	RegFile	A1	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]			IR_D[rs]		
D级功能部件	A2		IR_D[rt]	IR_D[rt]	IR_D[rt]		IR_D[rt]							
	D1													
	CMP				RF_RD1									
	D2				RF_RD2									
	EXT					IR_D[i16]	IR_D[i16]	IR_D[i16]	IR_D[i16]					
E级更新PC	NPC	PC4			PC4_D					PC4_D	PC4_D			
	index				IR_D[i16]					IR_D[i26]	IR_D[i26]		NPC_MUX	NPCS+1
	PC				NPC							RF_RD1	PC2_MUX	PC2S+1
D/E级流水线寄存器	IR_E		IR_D	IR_D		IR_D	IR_D	IR_D	IR_D		IR_D			
	PC4_E													
	PCB_E										PC4_D			
	RS_E		RF_RD1	RF_RD1		RF_RD1	RF_RD1	RF_RD1	RF_RD1					
	RT_E		RF_RD2	RF_RD2		RF_RD2								
	EXT_E					EXT	EXT	EXT	EXT					
E级功能部件	ALU	A	RS_E	RS_E		RS_E	RS_E	RS_E	RS_E					
	B		RT_E	RT_E		EXT_E	EXT_E	EXT_E	EXT_E				ALUB_MUX	ALUBSel
	XALU	D1												
E/M级流水线寄存器	D2													
	IR_M		IR_E	IR_E		IR_E	IR_E	IR_E	IR_E		IR_E			
	PC4_M											PC4_E		
	PCB_M													
	ALUout_M		ALUout	ALUout		ALUout	ALUout	ALUout	ALUout					
M级功能部件	XALUout_M													
	RT_M						RT_E							
	DM	A				ALUout_M	ALUout_M							
W级流水线寄存器	FD					RT_M								
	IR_W		IR_M	IR_M		IR_M		IR_M	IR_M		IR_M			
	PC4_W											PC4_M		
	PCB_W													
	ALUout_W		ALUout_M	ALUout_M				ALUout_M	ALUout_M					
W级功能部件	XALUout_W													
	DM_W					DM								
	EXT_DM	A												
F级功能部件	Din													
	A2		IR_W[rd]	IR_W[rd]		IR_W[rt]		IR_W[rt]	IR_W[rt]		0x1f		WReg_MUX	WRegSel
	RF	FD	ALUout_W	ALUout_W		DM_W		ALUout_W	ALUout_W		PC4_W		FD_MUX	FDSel

1. IF 级组合逻辑：

(1) PC.V

模块定义：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 0：无效 1：有效
stall	I	阻塞/暂停信号： 0：pc=npc 1：pc 保持不变
npc[31:0]	I	输入的 PC 地址
pc	O	输出当前 PC 地址

功能定义：

序号	功能	功能定义
1	复位	当时钟上升沿来临时，若复位信号有效， PC=0x00003000
2	取地址	时钟上升沿来临输出读取地址

(2) IM.V

容量为 16KB(32bit/word×4096word)

模块定义：

信号名	方向	描述
pc[31:0]	I	当前 PC 地址
Instr[31:0]	O	当前读取的指令

(3) ADD4.v

模块定义：

信号名	方向	描述
pc[31:0]	I	当前 pc 地址
pcplus4[31:0]	O	输出数据为地址加 4

(4) ADD8.v

模块定义：

信号名	方向	描述
pc[31:0]	I	当前 pc 地址
Pcplus8[31:0]	O	输出数据为地址加 8

2. IF/ID 级流水寄存器：

IF\_ID\_register.v

模块定义：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 0:无效 1:有效
en	I	写使能信号 0:不可写流水寄存器 1:可写流水寄存器
IR_D_in[31:0]	I	传入该寄存器的指令
PC4_D_in[31:0]	I	传入该寄存器的 PC+4
PC8_D_in[31:0]	I	传入该寄存器的 PC+8
IR_D_out[31:0]	O	传出该寄存器的指令
PC4_D_out[31:0]	O	传出该寄存器的 PC+4
PC8_D_out[31:0]	O	传出该寄存器的 PC+8

功能定义：

序号	功能	功能定义
1	复位	当时钟上升沿来临时，若复位信号有效，寄存器内容全为零
2	取地址	时钟上升沿来临时输出读取地址
3	取指令	时钟上升沿来临时取出当前指令

3. ID 级组合逻辑：

（1）GRF.v

模块定义：

信号名	方向	描述
-----	----	----

clk	I	时钟信号
reset	I	复位信号，将 32 个寄存器中的值全部清零 1：有效 0：无效
pc[31:0]	I	W 级 PC 地址（PC4_W-4）
RegWrite_W	I	W 级写使能信号 1：可向 GRF 中写入数据 0：不能向 GRF 中写入数据
Read_register1[4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个， 将其中存储的数据读出到 D1
Read_register2[4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个， 将其中存储的数据读出到 D2
Write_register_W	I	5 位地址输入信号，指定 32 个寄存器中的一个 作为写入的目标寄存器
Write_data_W[31:0]	I	向写入寄存器写入的数据

功能定义：

序号	功能名称	功能描述
1	复位	reset 信号有效时，所有寄存器存储的数值清零
2	读数据	读出 Read_register1, Read_register2 地址对应寄存器中所存储的数据到 RF. RD1, RF. RD2
3	写数据	当 WE 有效且时钟上升沿来临时，将 Write_data_W 写入 Write_register_W 所对应的寄存器中

## （2）EXT.v:

功能：选择立即数扩展方式

模块定义：

信号名	方向	描述
-----	----	----

imm[15:0]	I	输入数据
Extop[1:0]	I	选择信号： 00：无符号扩展 01：有符号扩展 10：加载至高位，低位补零
after_ext[31:0]	O	符号扩展后输出数据

### (3) CMP.v

功能：比较器

模块定义：

信号名	方向	描述
D1[31:0]	I	第一个比较的数
D2[31:0]	I	第二个比较的数
equal	O	判断信号 1: D1=D2 0:D1!=D2
g_or_e	O	判断信号 1: D1>=D2 0:D1<D2
greater	O	判断信号 1: D1>D2 0:D1<=D2

### (4) PC\_beq.v

模块定义：

信号名	方向	描述
after_ext[31:0]	I	EXT 扩展后的数
PC4_D[31:0]	I	PC+4 的值

equal	I	相等信号
g_or_e	I	大于等于信号
greater	I	大于信号
pc_beq	O	B 类指令跳转地址

#### (5) PC\_jal.v

模块定义：

信号名	方向	描述
Instr[31:0]	I	指令
PC4_D[31:0]	I	PC+4 的值
pc_jal	O	jal 指令跳转地址

#### (6) MFRSD.v

功能：D 级 rs 转发多选器

模块定义：

信号名	方向	描述
RF_RD1[31:0]	I	rs 寄存器里面内容
ALUout_M[31:0]	I	M 级 ALUout 数据
Write_data_W[31:0]	I	W 级多选器的输出内容
ForwardRSD[1:0]	I	选择信号 00: RF_RD1_trans=RF_RD1 01: RF_RD1_trans=ALUout_M 10: RF_RD1_trans=Write_data_W
RF_RD1_trans[31:0]	O	选择出来的数据

#### (7) MFRTD.v

功能：D 级 rt 转发多选器

模块定义：

信号名	方向	描述
RF_RD2[31:0]	I	rs 寄存器里面内容
ALUout_M[31:0]	I	M 级 ALUout 数据
Write_data_W[31:0]	I	W 级多选器的输出内容
ForwardRTD[1:0]	I	选择信号 00: RF_RD2_trans=RF_RD2 01: RF_RD2_trans=ALUout_M 10: RF_RD2_trans=Write_data_W
RF_RD2_trans[31:0]	O	选择出来的数据

#### (8) nextpc\_2.v

功能：跳转 pc 的选择

模块定义：

信号名	方向	描述
pc_jal[31:0]	I	jal 跳转的地址
pc_beq[31:0]	I	beq 跳转的地址
RF_RD1_trans[31:0]	I	jr 跳转的地址
pc_sel2[1:0]	I	选择信号 00: nextpc=pc_jal 01: nextpc=pc_beq 10: nextpc=RF_RD1_trans
nextpc[31:0]	O	选择出来的 nextpc

#### (9) Wreg\_D.v

模块定义：

信号名	方向	描述
Instr[20:16]	I	rt 寄存器
Instr[15:11]	I	rd 寄存器
5'b11111	I	31 号 (\$ra) 寄存器



RegDst[1:0]	I	写寄存器选择信号 00: write_register_D=rt 01: write_register_D=rd 10: write_register_D=\$ra
write_register_D[4:0]	O	选择出来的写寄存器

#### 4. ID/EX 级流水寄存器：

ID\_EX\_register.v

模块定义：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 0:无效 1: 有效
stall	I	阻塞/暂停信号
IR_E_in[31:0]	I	传入该寄存器的指令
PC4_E_in[31:0]	I	传入该寄存器的 PC+4
PC8_E_in[31:0]	I	传入该寄存器的 PC+8
RS_E_in[31:0]	I	由 rs 寄存器传出，传入该寄存器的值
RT_E_in[31:0]	I	由 rt 寄存器传出，传入该寄存器的值
EXT_E_in[31:0]	I	传入该寄存器的立即数扩展之后的值
write_register_E_in[4:0]	I	传入该寄存器的写寄存器
Start_E_in	I	传入该寄存器的触发信号
IR_E_out[31:0]	O	传出该寄存器的指令
PC4_E_out[31:0]	O	传出该寄存器的 PC+4
PC8_E_out[31:0]	O	传出该寄存器的 PC+8
RS_E_out[31:0]	O	由 rs 寄存器传出，传出该寄存器的值
RT_E_out[31:0]	O	由 rt 寄存器传出，传出该寄存器的值

EXT_E_out[31:0]	O	传出该寄存器的立即数扩展之后的值
write_register_E_out[4:0]	O	传出该寄存器的写寄存器
Start_E_out	O	传出该寄存器的触发信号

## 5. EX 级组合逻辑：

### (1) ALU\_data\_B.v

功能：选择进入 ALU 的第二个数据值

模块定义：

信号名	方向	描述
RF_RD2_trans[31:0]	I	经过转发选择后的 1 寄存器的值
after_ext[31:0]	I	立即数扩展之后的值
ALUSrc	I	选择信号
ALUB	O	输入 ALU 的第二个数据值

### (2) ALU.v

模块定义：

信号名	方向	描述
A[31:0]	I	输入 A 数据
B[31:0]	I	输入 B 数据
shamt[4:0]	I	输入的 5 位移位的位数
ALUop[3:0]	I	选择信号： 4'b0000:Result=A+B; 4'b0001:Result=A-B; 4'b0010:Result=A B; 4'b0011:Result=A&B; 4'b0100:Result=A^B; 4'b0101:Result=~(A B);

		4' b0110:Result=B<<shamt; 4' b0111:Result=B<<A[4:0]; 4' b1000:Result=B>>shamt; 4' b1001:Result=B>>A[4:0]; 4' b1010:Result=\$signed(B)>>>shamt; 4' b1011:Result=\$signed(B)>>>A[4:0]; 4' b1100:Result=(\$signed(A)<\$signed(B)) ?32' b1:32' b0; 4' b1101:Result=(A<B)?32' b1:32' b0;
Result[31:0]	O	计算后输出数据

### (3) MFRSE.v

功能：E 级转发多选器

模块定义：

信号名	方向	描述
RF_RD1[31:0]	I	第一个寄存器传出来的值
ALUout_M[31:0]	I	M 级 ALUout 数据
Write_data_W[31:0]	I	W 级多选器的输出内容
ForwardRSE[1:0]	I	选择信号 00: RF_RD1_trans=RF_RD1 01: RF_RD1_trans=ALUout_M 10: RF_RD1_trans=Write_data_W
RF_RD1_trans[31:0]	O	rs 转发多选器选出来的值

### (4) MFRTE.v

功能：E 级转发多选器

模块定义：

信号名	方向	描述
RF_RD2[31:0]	I	第二个寄存器传出来的值
ALUout_M[31:0]	I	M 级 ALUout 数据
Write_data_W[31:0]	I	W 级多选器的输出内容
ForwardRTE[1:0]	I	选择信号 00: RF_RD2_trans=RF_RD2 01: RF_RD2_trans=ALUout_M 10: RF_RD2_trans=Write_data_W
RF_RD2_trans[31:0]	O	rt 转发多选器选出来的值

#### (5) XALU.v

功能：有限状态机

模块定义：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
Instr[31:0]	I	31 位指令
D1	I	输入的第一个乘除数据
D2	I	输入的第二个乘除数据
Start	I	乘除类指令开始信号
HI[31:0]	O	HI 寄存器中的值
LO[31:0]	O	LO 寄存器中的值
Busy	O	阻塞信号

## 6. EX/MEM 级流水寄存器:

EX\_MEM\_register.v

模块定义:

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 0:无效 1: 有效
IR_M_in[31:0]	I	传入该寄存器的指令
PC4_M_in[31:0]	I	传入该寄存器的 PC+4
PC8_M_in[31:0]	I	传入该寄存器的 PC+8
ALUout_M_in[31:0]	I	由 ALU 传出, 传入该寄存器的值
RT_M_in[31:0]	I	由 rt 寄存器传出, 传入该寄存器的值
RegWrite_M_in	I	传入该寄存器的写信号
write_register_M_in[4:0]	I	传入该寄存器的写寄存器
IR_M_out[31:0]	O	传出该寄存器的指令
PC4_M_out[31:0]	O	传出该寄存器的 PC+4
PC8_M_out[31:0]	O	传出该寄存器的 PC+8
ALUout_M_out[31:0]	O	由 ALU 传出, 传出该寄存器的值
RT_M_out[31:0]	O	由 rt 寄存器传出, 传出该寄存器的值
RegWrite_M_out	O	传出该寄存器的写信号
write_register_M_out[4:0]	O	传出该寄存器的写寄存器

## 7. MEM 级组合逻辑

(1) DM.v

功能: 对内存进行读写操作

容量为 16KB(32bit/word×4096word)

模块定义：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 0: 无效 1: 有效
pc[31:0]	I	pc 现在地址
addr[31:0]	I	存数据的地址
MemWrite	I	写内存信号 0: 不可写内存 1: 可写内存
MemData[31:0]	I	存入的数据
BE[3:0]	I	字节有效信号
ext_op	I	扩展方式信号
DMout[31:0]	O	读出对应内存位置的数据

功能定义：

序号	功能名称	功能描述
1	读数据	读出 pc 地址对应内存中所存储的数据到 DMout
2	写数据	当 MemWrite 有效且时钟上升沿来临时，将 MemData 写入 addr 所对应的内存位置

(2) MFRTM.v

功能：M 级 rt 转发选择器

模块定义：

信号名	方向	描述
WD[31:0]	I	M 级 ALUout_M 的数据
Write_data_W[31:0]	I	W 级多选器的输出内容

ForwardRTM	I	选择信号 0: Write_data_trans=WD 1: Write_data_trans=Write_data_W
Write_data_trans[31:0]	O	输出传至 DMWD 端口数据

### (3) ByteEnable.v

功能：判断哪些字节有效

模块定义：

信号名	方向	描述
s_l_op[1:0]	I	指令信号 01: sw、lw 10: sh、lh、lhu 11---sb、lb、lbu
addr[31:0]	I	31 位指令
BE[3:0]	O	输出哪些字节有效

## 8. MEM/WB 级流水寄存器：

MEM\_WB\_register.v

模块定义：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 0:无效 1: 有效
IR_W_in[31:0]	I	传入该寄存器的指令
PC4_W_in[31:0]	I	传入该寄存器的 PC+4
PC8_W_in[31:0]	I	传入该寄存器的 PC+8
ALUout_W_in[31:0]	I	由 ALU 传出，传入该寄存器的值
DM_W_in[31:0]	I	由 DM 传出，传入该寄存器的值

RegWrite_W_in	I	传入该寄存器的写信号
write_register_W_in[4:0]	I	传入该寄存器的写寄存器
IR_W_out[31:0]	O	传出该寄存器的指令
PC4_W_out[31:0]	O	传出该寄存器的 PC+4
PC8_W_out[31:0]	O	传出该寄存器的 PC+8
ALUout_W_out[31:0]	O	由 ALU 传出，传出该寄存器的值
DM_W_out[31:0]	O	由 DM 传出，传出该寄存器的值
RegWrite_W_out	O	传出该寄存器的写信号
write_register_W_out[4:0]	O	传出该寄存器的写寄存器

## 9. WB 级组合逻辑：

### (1) DATAtoREG.v

功能：选择回写寄存器堆的数据来源

模块定义：

信号名	方向	描述
ALUout_W[31:0]	I	从 ALU 出来的数据
DMout[31:0]	I	从 DM 出来的数据
MemtoReg	I	选择信号： 0: writeback_data=ALUout_W 1: writeback_data=DMout
writeback_data[31:0]	O	输出数据作为回写寄存器内容

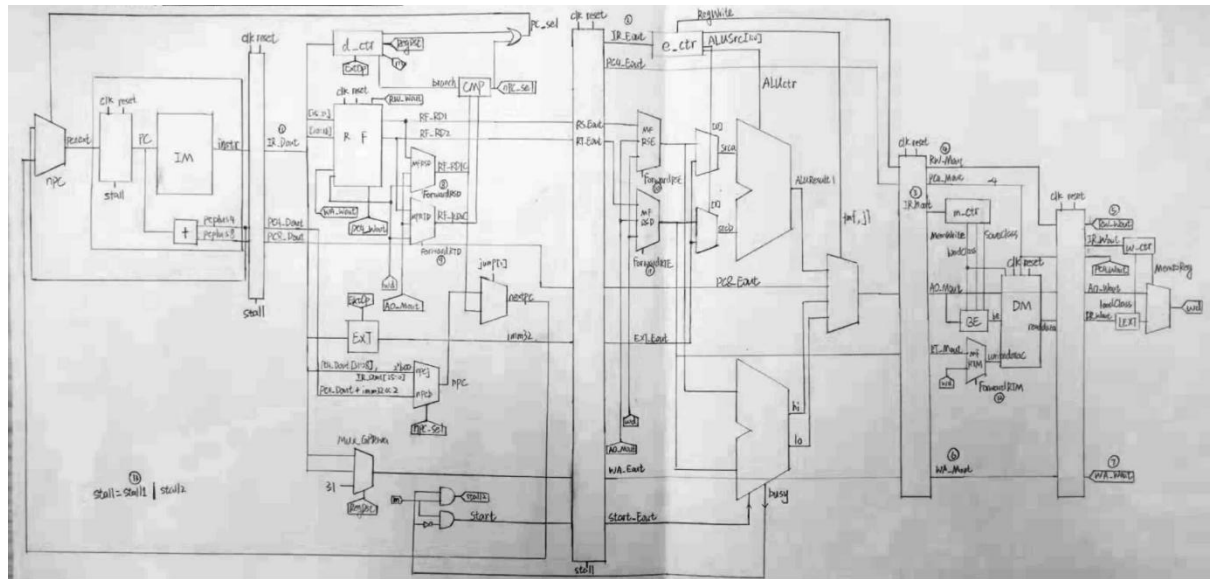
## 三. 控制器

模块定义：


信号名	方向	描述
func[5:0]	I	6 位 func
op[5:0]	I	6 位 op



RegDst[1:0]	O	写寄存器选择信号
ALUSrc	O	进入 ALU 的第二个值选择信号
MemtoReg	O	写回写寄存器的数据选择信号
RegWrite	O	写入寄存器信号
MemWrite	O	写入 DM 信号
Extop[1:0]	O	位扩展信号
ALUOp[1:0]	O	ALU 功能选择信号
pc_sel1	O	是否跳转信号
pc_sel2[1:0]	O	哪种跳转方式选择信号
Cal_r	O	Cal_r 类信号
Cal_i	O	Cal_i 类信号
B	O	Beq 类信号
Load	O	Load 类信号
Save	O	Save 类信号
J	O	J 类信号
mf_type[1:0]	O	mf 类型指令信号 00:非 mf 类型 01: 对应 HI 10: 对应 LO
s_l_op[1:0]	O	存取指令的类型信号
ext_op	O	存取指令进行扩展方式 0: 0 扩展 1: 符号扩展



### 三. 转发和暂停

						ID/EX(Tnew)	EX/MEM(Tnew)				MEM/WB(Tnew)							
流水级	源寄存器	涉及指令				jal 0/31	cal_r 0/rd	cal_i 0/rt	jal 0/31	cal_r 0/rd	cal_i 0/rt	lw 0/rt	jal 0/31					
IF/ID (D)	rs	beq, jr	MFRSD	ForwardRSD	RF_RD1	ADD4	ALUout	ALUout	ADD4	RFRD MUX	RFRD MUX	RFRD MUX						
	rt	beq	MFRTD	ForwardRTD	RF_RD2		ALUout	ALUout		RFRD MUX	RFRD MUX	RFRD MUX						
ID/EX (E)	rs	cal r, cal i, lw, sw	MFRSE	ForwardRSE	RSSE		ALUout	ALUout		RFRD MUX	RFRD MUX	RFRD MUX						
	rt	cal r	MFRTE	ForwardRTE	RTSE		ALUout	ALUout		RFRD MUX	RFRD MUX	RFRD MUX						
EX/MEM (M)	rt	sw	MFRTM	ForwardRTM	RTOM					RFRD MUX	RFRD MUX	RFRD MUX						
			转发MUX	控制信号	输入0													

IF/ID当前指令			ID/EX (Tnew)			EX/MEM (Tnew)
指令类型	源寄存器	Tuse	cal_r 1/rd	cal_i 1/rt	lw 2/rt	lw 1/rt
cal_r	rs/rt	1			暂停	
cal_i	rs	1			暂停	
lw	rs	1			暂停	
sw	rt	2			暂停	
beq	rs/rt	0	暂停	暂停	暂停	暂停
jr	rs	0	暂停	暂停	暂停	暂停
jalr	rs	0	暂停	暂停	暂停	暂停

对于暂停信号 stall

stall\_B\_Calr=(B\_D&&Cal\_r\_E&&((IR\_D\_out[25:21]==IR\_E\_out[15:11])|(IR\_D\_out[20:16]==IR\_E\_out[15:11])));

stall\_B\_Cali=(B\_D&&Cal\_i\_E&&((IR\_D\_out[25:21]==IR\_E\_out[20:16])|(IR\_D\_out[20:16]==IR\_E\_out[20:16])));

stall\_B\_Load1=(B\_D&&Load\_E&&((IR\_D\_out[25:21]==IR\_E\_out[20:16])|(IR\_D\_out[20:16]==IR\_E\_out[20:16])));

stall\_B\_Load2=(B\_D&&Load\_M&&((IR\_D\_out[25:21]==IR\_M\_out[20:16])|(IR\_D\_out[20:16]==IR\_M\_out[20:16])));

```
stall_Calr_Load=(Cal_r_D&&Load_E&&((IR_D_out[25:21]==IR_E_out[20:16])|(IR_D_out[20:16]==IR_E_out[20:16])));
```

```
stall_Cali_Load=(Cal_i_D&&Load_E&&(IR_D_out[25:21]==IR_E_out[20:16]));
```

```
stall_Load_Load=(Load_D&&Load_E&&(IR_D_out[25:21]==IR_E_out[20:16]));
```

```
stall_Save_Load=(Save_D&&Load_E&&(IR_D_out[25:21]==IR_E_out[20:16]));
```

```
stall=stall_m|stall_B_Calr|stall_B_Cali|stall_B_Load1|stall_B_Load2|stall_Calr_Load|stall_Cali_Load|stall_Load_Load|stall_Save_Load;
```

#### 四. 思考题

1. 为什么需要有单独的乘除法部件而不是整合进 ALU? 为何需要有独立的 HI、LO 寄存器?

答: 乘除法运算需要延迟, 我们假定乘/除部件的执行乘法的时间为 5 个 cycle, 执行除法的时间为 10 个 cycle, 在乘除部件需要模拟这个延迟, 这会导致 CPU 运算乘除法延迟时间较长。除此之外, 乘除法运算结果会超出 32 位。为了提高 CPU 效率, 在进行乘除法时不影响其他指令执行, 因此需要单独设立模块。

HI 和 LO 寄存器不能被编码。它们在乘除法时被使用, 计算结果的不同位储存在不同的寄存器中。避免在简单流水线中从高延迟指令回写寄存器文件的问题。例如: 当 multu 产生计算值后, 要写入两个值, 与此同时也有可能流水线其他指令也要写入寄存器, 这样会产生冲突。而建立 HI、LO 寄存器, 则可以避免冲突。

2. 参照你对延迟槽的理解, 试解释“乘除槽”。

答: 引入延时槽的主要目的是提高流水线的效率。跟延迟槽类似, 乘除槽是指在乘除法运算时, 和 HI 或 LO 寄存器相关的指令全部被冻结在 ID 级, 而其他的指令可以顺利向前进行, 不会被冻结阻塞。这样可以保证 HI 和 LO 寄存器的值稳定。

3. 为何上文文末提到的 lb 等指令使用的数据扩展模块应在 MEM/WB 之后, 而不

能在 DM 之后？

答：放在 DM 之后一定会使该级延迟增加一个拓展器时间，可能会导致紊乱而时序出错。而放在 MEM/WB 之后对于一些不需要写寄存器的指令，相对于前一种情况，延时减少了这个拓展器时间。

4. 举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。

（Hint：考虑 C 语言中字符串的情况）

答：对于 C 语言中字符串的情况，如果是按照字节寻址，则可以直接找到地址，可以是不是 4 的倍数的字符。而如果按照字寻址，则还需要首先将寻找地址右移两位（即除以 4）后细找，增加了运算时间而且操作起来更加麻烦，这时按字节访问内存相对于按字访问内存性能上更有优势。

5. 如何概括你所设计的 CPU 的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？

答：DETECTOR 型。hazard 单元像是一位各种冲突的侦测者，它本身遵循着自己的一套逻辑，根据对不同阶段流水指令的解析结果，判断出哪个寄存器同时有读入和输出（数据冒险），是否会对程序控制产生影响（控制冒险）。

在转发操作时并没有将指令进行类型划分，而是在代码级对流水线的冒险控制进行了处理。在暂停操作，通过指令类型划分，得出暂停信号的表达。

6. 你对流水线 CPU 设计风格有何见解？

（如果你觉得你的思考值得分享，不妨请在讨论发表你自己的观点和文章，我们会从中发掘优秀文本以飨后辈并予以分数上的鼓励。）

答：这种写法控制代码短小精悍且具有一定的代表性。同段代码适用于多类指令，面对添加指令，很多时候不用修改代码，避免了很多麻烦也降低了出错概率。除此之外，这种方式从冲突产生的原因着手，从原理上理解。

7. 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。（非常重要）

(1) lw 后加 beq 指令出现异常，lw 后加 beq 应该暂停两次。

测试程序：

```
ori $a0, $0, 2026
sw $a0, 0($0)
lw $a1, 0($0)
lw $a2, 0($0)
beq $a2, $a1, branch
ori $t0, $0, 1
ori $t0, $0, 2
branch:
ori $t1, $0, 3
```

预期结果：

```
35@00003000: $ 4 <= 000007ea
35@00003004: *00000000 <= 000007ea
55@00003008: $ 5 <= 000007ea
65@0000300c: $ 6 <= 000007ea
105@00003014: $ 8 <= 00000001
115@0000301c: $ 9 <= 00000003
```

(2) jr 在 D 级需要用到 rs 寄存器的值，需要通过转发来解决，不然中强测前两个点过不去。

测试程序：

```
ori $ra, $0, 0x3014
jr $ra
ori $a0, $0, 1
ori $a0, $0, 2
ori $a0, $0, 3
ori $a0, $0, 4
```

预期结果：

```
35@00003000: $31 <= 00003014
65@00003008: $ 4 <= 00000001
75@00003014: $ 4 <= 00000004
```

(3) 在 D 级部件将通过转发后的 RF\_RD1\_trans 和 RF\_RD2\_trans 转发到了下一级寄存器中，实际上应该将转发前的 RF\_RD1 和 RF\_RD2 传到下一级。

(4) jal 延迟槽后跟 jr 发生冲突，在 E 级增加多路选择器，，选择传出的地址是 ALU 计算出来的值还是 PC+8

测试程序：

```
jal jump
nop
jump:
jr $ra
ori $a0,$0,1
```

预期结果：

```
-----
35@00003000: $31 <= 00003008
65@0000300c: $ 4 <= 00000001
85@0000300c: $ 4 <= 00000001
105@0000300c: $ 4 <= 00000001
125@0000300c: $ 4 <= 00000001
145@0000300c: $ 4 <= 00000001
165@0000300c: $ 4 <= 00000001
185@0000300c: $ 4 <= 00000001
205@0000300c: $ 4 <= 00000001
225@0000300c: $ 4 <= 00000001
245@0000300c: $ 4 <= 00000001
265@0000300c: $ 4 <= 00000001
285@0000300c: $ 4 <= 00000001
305@0000300c: $ 4 <= 00000001
325@0000300c: $ 4 <= 00000001
345@0000300c: $ 4 <= 00000001
365@0000300c: $ 4 <= 00000001
385@0000300c: $ 4 <= 00000001
405@0000300c: $ 4 <= 00000001
425@0000300c: $ 4 <= 00000001
445@0000300c: $ 4 <= 00000001
465@0000300c: $ 4 <= 00000001
485@0000300c: $ 4 <= 00000001
505@0000300c: $ 4 <= 00000001
525@0000300c: $ 4 <= 00000001
545@0000300c: $ 4 <= 00000001
565@0000300c: $ 4 <= 00000001
585@0000300c: $ 4 <= 00000001
605@0000300c: $ 4 <= 00000001
625@0000300c: $ 4 <= 00000001
645@0000300c: $ 4 <= 00000001
665@0000300c: $ 4 <= 00000001
```

(5) beq 指令当跳转条件不满足时，应传回 PC+8 的值，而不是 PC+4，不然中强测第二个点过不去。

测试程序：

```

ori $a0,$0,1
ori $a1,$0,2
beq $a0,$a1,jump
ori $t0,$0,3
ori $t1,$0,4
addu $t7,$a0,$a1
jump:
subu $t7,$a1,$a0

```

预期结果:

```

35@00003000: $ 4 <= 00000001
45@00003004: $ 5 <= 00000002
75@00003008: $ 8 <= 00000003
85@00003010: $ 9 <= 00000004
95@00003014: $15 <= 00000003
105@00003018: $15 <= 00000001

```

## 五. 测试程序

X-Y-Z

X: 冲突前序指令类型

Y: 后续指令位于 D 级时，前序指令的级数

Z: 产生冲突的寄存器

**Addu:**

**Ori\_E\_RS(addu):**

```

ori $t0,$zero,8
addu $t1,$t0,$zero
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 00000008

```

**Ori\_M\_RS(addu)**

```

ori $t0,$zero,8
ori $t2,$zero,12
addu $t1,$t0,$zero
180@00003000: $ 8 <= 00000008

```

220@00003004: \$10 <= 0000000c  
260@00003008: \$ 9 <= 00000008

#### **Ori\_W\_RS(addu)**

ori \$t0, \$zero, 8  
ori \$t2, \$zero, 12  
ori \$t3, \$zero, 13  
addu \$t1, \$t0, \$zero  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$10 <= 0000000c  
260@00003008: \$11 <= 0000000d  
300@0000300c: \$ 9 <= 00000008

#### **Ori\_E\_RT(addu)**

ori \$t0, \$zero, 8  
addu \$t1, \$zero, \$t0  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$ 9 <= 00000008

#### **Ori\_M\_RT(addu)**

ori \$t0, \$zero, 8  
ori \$t2, \$zero, 12  
addu \$t1, \$zero, \$t0  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$10 <= 0000000c  
260@00003008: \$ 9 <= 00000008

#### **Ori\_W\_RT(addu)**

ori \$t0, \$zero, 8  
ori \$t2, \$zero, 12  
ori \$t3, \$zero, 13  
addu \$t1, \$zero, \$t0  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$10 <= 0000000c  
260@00003008: \$11 <= 0000000d  
300@0000300c: \$ 9 <= 00000008



**Lui\_E\_RS(addu)**

lui \$t0, 8  
addu \$t1, \$t0, \$zero  
180@00003000: \$ 8 <= 00080000  
220@00003004: \$ 9 <= 00080000

**Lui\_M\_RS(addu)**

lui \$t0, 8  
lui \$t2, 12  
addu \$t1, \$t0, \$zero  
180@00003000: \$ 8 <= 00080000  
220@00003004: \$10 <= 000c0000  
260@00003008: \$ 9 <= 00080000

**Lui\_W\_RS(addu)**

lui \$t0, 8  
lui \$t2, 12  
lui \$t3, 14  
addu \$t1, \$t0, \$zero  
180@00003000: \$ 8 <= 00080000  
220@00003004: \$10 <= 000c0000  
260@00003008: \$11 <= 000e0000  
300@0000300c: \$ 9 <= 00080000

**Lui\_W\_RT(addu)**

lui \$t0, 8  
addu \$t1, \$zero, \$t0  
180@00003000: \$ 8 <= 00080000  
220@00003004: \$ 9 <= 00080000

**Lui\_w\_RT(addu)**

lui \$t0, 8  
lui \$t2, 12  
addu \$t1, \$zero, \$t0  
180@00003000: \$ 8 <= 00080000  
220@00003004: \$10 <= 000c0000  
260@00003008: \$ 9 <= 00080000

**Lui\_W\_RT(addu)**

```
lui $t0, 8
lui $t2, 12
lui $t3, 14
addu $t1, $zero, $t0
180@00003000: $ 8 <= 00080000
220@00003004: $10 <= 000c0000
260@00003008: $11 <= 000e0000
300@0000300c: $ 9 <= 00080000
```

#### **R\_E\_RS(addu)**

```
lui $t0, 8
addu $t1, $zero, $t0
addu $t2, $t1, $zero
180@00003000: $ 8 <= 00080000
220@00003004: $ 9 <= 00080000
260@00003008: $10 <= 00080000
```

#### **R\_M\_RS(addu)**

```
lui $t0, 8
addu $t1, $zero, $t0
addu $t3, $t0, $t0
addu $t2, $t1, $zero
180@00003000: $ 8 <= 00080000
220@00003004: $ 9 <= 00080000
260@00003008: $11 <= 00100000
300@0000300c: $10 <= 00080000
```

#### **R\_W\_RS(RT)(addu)**

```
lui $t0, 8
addu $t1, $zero, $t0
addu $t3, $t0, $t0
addu $t4, $t0, $t0
addu $t2, $t1, $t1
180@00003000: $ 8 <= 00080000
220@00003004: $ 9 <= 00080000
260@00003008: $11 <= 00100000
300@0000300c: $12 <= 00100000
340@00003010: $10 <= 00100000
```

#### **Ld\_E\_RS(RT)(addu)**

```
ori $t0, $zero, 8
ori $t1, $zero, 12
ori $t2, $zero, 16
ori $t3, $zero, 20
```

```

ori $t4, $zero, 24
ori $t5, $zero, 28
sw $t1, 0($t0)
ori $s0, $zero, 4
ori $s1, $zero, 8
ori $s2, $zero, 12
lw $t6, 0($t0)
addu $t7, $t6, $t6
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 0000000c
260@00003008: $10 <= 00000010
300@0000300c: $11 <= 00000014
340@00003010: $12 <= 00000018
380@00003014: $13 <= 0000001c
380@00003018: *00000008 <= 0000000c
460@0000301c: $16 <= 00000004
500@00003020: $17 <= 00000008
540@00003024: $18 <= 0000000c
580@00003028: $14 <= 0000000c
660@0000302c: $15 <= 00000018

```

### **Ld\_M\_RS(RT)(addu)**

```

ori $t0, $zero, 8
ori $t1, $zero, 12
ori $t2, $zero, 16
ori $t3, $zero, 20
ori $t4, $zero, 24
sw $t1, 0($t0)
ori $s0, $zero, 4
ori $s1, $zero, 8
ori $s2, $zero, 12
lw $t6, 0($t0)
ori $t5, $zero, 28
addu $t7, $t6, $t6
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 0000000c
260@00003008: $10 <= 00000010
300@0000300c: $11 <= 00000014
340@00003010: $12 <= 00000018
340@00003014: *00000008 <= 0000000c
420@00003018: $16 <= 00000004
460@0000301c: $17 <= 00000008
500@00003020: $18 <= 0000000c
540@00003024: $14 <= 0000000c

```

580@00003028: \$13 <= 0000001c  
620@0000302c: \$15 <= 00000018

### **Ld\_W\_RS(RT)(addu)**

ori \$t0, \$zero, 8  
ori \$t1, \$zero, 12  
ori \$t2, \$zero, 16  
ori \$t3, \$zero, 20  
ori \$t4, \$zero, 24  
sw \$t1, 0(\$t0)  
ori \$s0, \$zero, 4  
ori \$s1, \$zero, 8  
ori \$s2, \$zero, 12  
lw \$t6, 0(\$t0)  
ori \$t5, \$zero, 28  
ori \$t8, \$zero, 32  
addu \$t7, \$t6, \$t6  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$ 9 <= 0000000c  
260@00003008: \$10 <= 00000010  
300@0000300c: \$11 <= 00000014  
340@00003010: \$12 <= 00000018  
340@00003014: \*00000008 <= 0000000c  
420@00003018: \$16 <= 00000004  
460@0000301c: \$17 <= 00000008  
500@00003020: \$18 <= 0000000c  
540@00003024: \$14 <= 0000000c  
580@00003028: \$13 <= 0000001c  
620@0000302c: \$24 <= 00000020  
660@00003030: \$15 <= 00000018

### **Jal\_E\_RS(RT)(addu)**

ori \$t0, \$zero, 8  
ori \$t1, \$zero, 12  
ori \$t2, \$zero, 16  
jal change1  
addu \$t3, \$ra, \$ra  
ori \$t4, \$zero, 20  
ori \$t5, \$zero, 24  
change1:  
ori \$t6, \$zero, 20  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$ 9 <= 0000000c  
260@00003008: \$10 <= 00000010

300@0000300c: \$31 <= 00003014  
340@00003010: \$11 <= 00006028  
380@0000301c: \$14 <= 00000014

### **Jal\_M\_RS(RT)(addu)**

```
ori $t0, $zero, 8
ori $t1, $zero, 12
ori $t2, $zero, 16
jal change1
ori $t4, $zero, 20
ori $t5, $zero, 24
change1:
    addu $t3, $ra, $ra
    ori $t6, $zero, 20
    ori $t7, $zero, 24
$ 8 <= 00000008
$ 9 <= 0000000c
$10 <= 00000010
$31 <= 00003014
$12 <= 00000014
$11 <= 00006028
$14 <= 00000014
$15 <= 00000018
```

### **Jal\_W\_RS(RT)(addu)**

```
ori $t0, $zero, 8
ori $t1, $zero, 12
ori $t2, $zero, 16
jal change1
ori $t4, $zero, 20
ori $t5, $zero, 24
change1:
    ori $t6, $zero, 20
    addu $t3, $ra, $ra
    ori $t6, $zero, 20
    ori $t7, $zero, 24
$ 8 <= 00000008
$ 9 <= 0000000c
$10 <= 00000010
$31 <= 00003014
$12 <= 00000014
$14 <= 00000014
$11 <= 00006028
```

\$14 <= 00000014

\$15 <= 00000018

### **Ori\_E\_RS(ori)**

ori \$t0, \$zero, 8

ori \$t1, \$t0, 12

180@00003000: \$ 8 <= 00000008

220@00003004: \$ 9 <= 0000000c

### **Ori\_M\_RS(ori)**

ori \$t0, \$zero, 8

ori \$t2, \$zero, 20

ori \$t1, \$t0, 12

180@00003000: \$ 8 <= 00000008

220@00003004: \$10 <= 00000014

260@00003008: \$ 9 <= 0000000c

### **Ori\_W\_RS(ori)**

ori \$t0, \$zero, 8

ori \$t2, \$zero, 20

ori \$t3, \$zero, 24

ori \$t1, \$t0, 12

180@00003000: \$ 8 <= 00000008

220@00003004: \$10 <= 00000014

260@00003008: \$11 <= 00000018

300@0000300c: \$ 9 <= 0000000c

### **Subu\_E\_RS(ori)**

ori \$t0, \$zero, 8

ori \$t1, \$zero, 20

ori \$t2, \$zero, 24

ori \$t3, \$zero, 12

ori \$t4, \$zero, 16

subu \$t5, \$t0, \$t1

ori \$t6, \$t5, 13

180@00003000: \$ 8 <= 00000008

220@00003004: \$ 9 <= 00000014

260@00003008: \$10 <= 00000018

300@0000300c: \$11 <= 0000000c

340@00003010: \$12 <= 00000010

380@00003014: \$13 <= ffffffff4

420@00003018: \$14 <= ffffffff4d

### **Subu\_M\_RS(ori)**

```

ori $t0, $zero, 8
ori $t1, $zero, 20
ori $t2, $zero, 24
ori $t3, $zero, 12
ori $t4, $zero, 16
subu $t5, $t0, $t1
ori $t7, $zero, 20
ori $t6, $t5, 13
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 00000014
260@00003008: $10 <= 00000018
300@0000300c: $11 <= 0000000c
340@00003010: $12 <= 00000010
380@00003014: $13 <= ffffffff4
420@00003018: $15 <= 00000014
460@0000301c: $14 <= ffffffff4

```

#### **Subu\_W\_RS(ori)**

```

ori $t0, $zero, 8
ori $t1, $zero, 20
ori $t2, $zero, 24
ori $t3, $zero, 12
ori $t4, $zero, 16
subu $t5, $t0, $t1
ori $t7, $zero, 20
ori $t8, $zero, 24
ori $t6, $t5, 13
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 00000014
260@00003008: $10 <= 00000018
300@0000300c: $11 <= 0000000c
340@00003010: $12 <= 00000010
380@00003014: $13 <= ffffffff4
420@00003018: $15 <= 00000014
460@0000301c: $24 <= 00000018
500@00003020: $14 <= ffffffff4

```

#### **Ld\_E\_RS(ori)**

```

ori $t0, $zero, 8
ori $t1, $zero, 20
ori $t2, $zero, 4
ori $t3, $zero, 12
ori $t4, $zero, 16
sw $t0, 0($t1)

```

```

lw $t5, 0($t1)
ori $t6, $t5, 13
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 00000014
260@00003008: $10 <= 00000004
300@0000300c: $11 <= 0000000c
340@00003010: $12 <= 00000010
340@00003014: *00000014 <= 00000008
420@00003018: $13 <= 00000008
500@0000301c: $14 <= 0000000d

```

### **Ld\_M\_RS(ori)**

```

ori $t0, $zero, 8
ori $t1, $zero, 20
ori $t2, $zero, 4
ori $t3, $zero, 12
ori $t4, $zero, 16
sw $t0, 0($t1)
lw $t5, 0($t1)
ori $t7, $zero, 20
ori $t6, $t5, 13
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 00000014
260@00003008: $10 <= 00000004
300@0000300c: $11 <= 0000000c
340@00003010: $12 <= 00000010
340@00003014: *00000014 <= 00000008
420@00003018: $13 <= 00000008
460@0000301c: $15 <= 00000014
500@00003020: $14 <= 0000000d

```

### **Ld\_W\_RS(ori)**

```

ori $t0, $zero, 8
ori $t1, $zero, 20
ori $t2, $zero, 4
ori $t3, $zero, 12
ori $t4, $zero, 16
sw $t0, 0($t1)
lw $t5, 0($t1)
ori $t7, $zero, 20
ori $t8, $zero, 24
ori $t6, $t5, 13
180@00003000: $ 8 <= 00000008

```



220@00003004: \$ 9 <= 00000014  
260@00003008: \$10 <= 00000004  
300@0000300c: \$11 <= 0000000c  
340@00003010: \$12 <= 00000010  
340@00003014: \*00000014 <= 00000008  
420@00003018: \$13 <= 00000008  
460@0000301c: \$15 <= 00000014  
500@00003020: \$24 <= 00000018  
540@00003024: \$14 <= 0000000d

### **Jal\_E\_RS(ori)**

ori \$t0, \$zero, 8  
ori \$t1, \$zero, 20  
jal change1  
ori \$t2, \$ra, 8  
ori \$t3, \$zero, 14  
change1:  
    ori \$t4, \$zero, 18  
ori \$t5, \$zero, 22  
ori \$t6, \$zero, 26  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$ 9 <= 00000014  
260@00003008: \$31 <= 00003010  
300@0000300c: \$10 <= 00003018  
340@00003014: \$12 <= 00000012  
380@00003018: \$13 <= 00000016  
420@0000301c: \$14 <= 0000001a

### **Jal\_M\_RS(ori)**

ori \$t0, \$zero, 8  
ori \$t1, \$zero, 20  
jal change1  
ori \$t2, \$zero, 4  
ori \$t3, \$zero, 14  
change1:  
    ori \$t4, \$ra, 18  
ori \$t5, \$zero, 22  
ori \$t6, \$zero, 26  
180@00003000: \$ 8 <= 00000008  
220@00003004: \$ 9 <= 00000014  
260@00003008: \$31 <= 00003010  
300@0000300c: \$10 <= 00000004  
340@00003014: \$12 <= 00003012

380@00003018: \$13 <= 00000016  
420@0000301c: \$14 <= 0000001a

### **Jal\_W\_RS(ori)**

```
ori $t0, $zero, 8
ori $t1, $zero, 20
jal change1
ori $t2, $zero, 4
ori $t3, $zero, 14
change1:
    ori $t7, $zero, 6
    ori $t4, $ra, 18
ori $t5, $zero, 22
ori $t6, $zero, 26
180@00003000: $ 8 <= 00000008
220@00003004: $ 9 <= 00000014
260@00003008: $31 <= 00003010
300@0000300c: $10 <= 00000004
340@00003014: $15 <= 00000006
380@00003018: $12 <= 00003012
420@0000301c: $13 <= 00000016
460@00003020: $14 <= 0000001a
```

### **Lw:**

#### **R\_E/M/W\_RS(lw)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 9
ori $t3, $zero, 12
sw $t1, 0($t0)
sw $t2, 4($t0)
sw $t3, 8($t0)
```

```
occasion1: #R_E_RS
    subu $t4, $t1, $t0
    lw $t5, 0($t4)
```

```
occasion2: #R_M_RS
    subu $t5, $t3, $t0
    ori $zero, $zero, 5
    lw $t6, 0($t5)
```

```

occasion3:  #R_W_RS
            addu $t6, $t0, $t1
            ori $s0, $zero, 12
            ori $s1, $zero, 16
            lw $t7, 0($t6)
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 00000009
$11 <= 0000000c
*00000004 <= 00000008
*00000008 <= 00000009
*0000000c <= 0000000c
$12 <= 00000004
$13 <= 00000008
$13 <= 00000008
$14 <= 00000009
$14 <= 0000000c
$16 <= 0000000c
$17 <= 00000010
$15 <= 0000000c

```

### **I\_E/M/W\_RS(lw)**

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 9
ori $t3, $zero, 12
sw $t1, 0($t0)
sw $t2, 4($t0)
sw $t3, 8($t0)

```

```

occasion1:  #R_E_RS
            ori $t4, $zero, 4
            lw $t5, 0($t4)

```

```

occasion2:  #R_M_RS
            ori $t5, $zero, 8
            ori $zero, $zero, 5
            lw $t6, 0($t5)

```

```

occasion3:  #R_W_RS
            ori $t6, $zero, 12
            ori $s0, $zero, 12
            ori $s1, $zero, 16

```

```

        lw $t7, 0($t6)
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 00000009
$11 <= 0000000c
*00000004 <= 00000008
*00000008 <= 00000009
*0000000c <= 0000000c
$12 <= 00000004
$13 <= 00000008
$13 <= 00000008
$14 <= 00000009
$14 <= 0000000c
$16 <= 0000000c
$17 <= 00000010
$15 <= 0000000c

```

#### **Ld\_E/M/W\_RS(lw)**

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
sw $t0, 0($zero)
sw $t1, 0($t0)
sw $t2, 0($t1)
sw $t3, 4($t1)

```

occasion1: #ld\_E\_RS

```

        lw $t4, 0($t0)
        lw $t5, 0($t4)

```

occasion2: #ld\_M\_RS

```

        lw $t5, -4($t0)
        addu $zero, $zero, $t1
        lw $t6, 0($t5)

```

occasion: #ld\_W\_RS

```

        lw $t6, 4($t0)
        ori $s0, $zero, 1
        ori $s1, $zero, 2
        lw $t7, 0($t6)
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 0000000c
$11 <= 00000010

```

```

*00000000 <= 00000004
*00000004 <= 00000008
*00000008 <= 0000000c
*0000000c <= 00000010
$12 <= 00000008
$13 <= 0000000c
$13 <= 00000004
$14 <= 00000008
$14 <= 0000000c
$16 <= 00000001
$17 <= 00000002
$15 <= 00000010

```

由于 DM 的容量只有 4KB，因此 31 号寄存器中的值不可能作为 lw 指令中的寻址

## Sw:

### R\_E/M/W\_RS/RT(sw)

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
ori $t4, $zero, 20

```

```

occasion1: #R_E_RS
    addu $t5, $t0, $t1
    sw $t2, 0($t5)

```

```

occasion2: #R_M_RS
    subu $t6, $t2, $t1
    ori $s0, $zero, 12
    sw $t3, 4($t6)

```

```

occasion3: #R_W_RS
    subu $t7, $t4, $t0
    ori $s1, $zero, 4
    ori $s2, $zero, 8
    sw $t4, 0($t7)

```

```

occasion4: #R_E_RT
    subu $t5, $t1, $t0

```

sw \$t5, 0(\$t0)

occasion5: #R\_M\_RT  
subu \$t5, \$t2, \$t0  
ori \$s0, \$zero, 2  
sw \$t5, 4(\$t2)

occasion6: #R\_W\_RT  
addu \$t6, \$t3, \$t4  
ori \$s0, \$zero, 1  
ori \$s1, \$zero, 2  
sw \$t6, 0(\$t1)

\$ 8 <= 00000004  
\$ 9 <= 00000008  
\$10 <= 0000000c  
\$11 <= 00000010  
\$12 <= 00000014  
\$13 <= 0000000c  
\*0000000c <= 0000000c  
\$14 <= 00000004  
\$16 <= 0000000c  
\*00000008 <= 00000010  
\$15 <= 00000010  
\$17 <= 00000004  
\$18 <= 00000008  
\*00000010 <= 00000014  
\$13 <= 00000004  
\*00000004 <= 00000004  
\$13 <= 00000008  
\$16 <= 00000002  
\*00000010 <= 00000008  
\$14 <= 00000024  
\$16 <= 00000001  
\$17 <= 00000002  
\*00000008 <= 00000024

**I\_E/M/W\_RS/RT(sw)**

ori \$t0, \$zero, 4  
ori \$t1, \$zero, 8  
ori \$t2, \$zero, 12  
ori \$t3, \$zero, 16  
ori \$t4, \$zero, 20

```
occasion1: #I_E_RS
    ori $t5, $zero, 12
    sw $t2, 0($t5)
```

```
occasion2: #I_M_RS
    ori $t6, $zero, 4
    ori $s0, $zero, 12
    sw $t3, 4($t6)
```

```
occasion3: #I_W_RS
    ori $t7, $zero, 16
    ori $s1, $zero, 4
    ori $s2, $zero, 8
    sw $t4, 0($t7)
```

```
occasion4: #I_E_RT
    lui $t5, 3
    sw $t5, 0($t0)
```

```
occasion5: #I_M_RT
    lui $t5, 1
    ori $s0, $zero, 2
    sw $t5, 4($t2)
```

```
occasion6: #I_W_RT
    ori $zero, $zero, 9
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    sw $zero, 0($t1)
```

```
$ 8 <= 00000004
```

```
$ 9 <= 00000008
```

```
$10 <= 0000000c
```

```
$11 <= 00000010
```

```
$12 <= 00000014
```

```
$13 <= 0000000c
```

```
*0000000c <= 0000000c
```

```
$14 <= 00000004
```

```
$16 <= 0000000c
```

```
*00000008 <= 00000010
```

```
$15 <= 00000010
```

```
$17 <= 00000004
```

```
$18 <= 00000008
```

```
*00000010 <= 00000014
```

```
$13 <= 00030000
```

\*00000004 <= 00030000  
\$13 <= 00010000  
\$16 <= 00000002  
\*00000010 <= 00010000  
\$16 <= 00000001  
\$17 <= 00000002  
\*00000008 <= 00000000

### **Ld\_E/M/W\_RS\_RT(sw)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
ori $t4, $zero, 20
sw $t0, 0($zero)
sw $t1, 0($t0)
sw $t2, 4($t0)
sw $t3, 8($t0)
occasion1: #ld_E_RS
    lw $t5, 0($t0)
    sw $t2, 0($t5)
```

```
occasion2: #ld_M_RS
    lw $t6, 0($t0)
    ori $s0, $zero, 12
    sw $t3, 4($t6)
```

```
occasion3: #ld_W_RS
    lw $t7, 4($t0)
    ori $s1, $zero, 4
    ori $s2, $zero, 8
    sw $t4, 0($t7)
```

```
occasion4: #ld_E_RT
    lw $t5, 0($t2)
    sw $t5, 0($t0)
```

```
occasion5: #ld_M_RT
    lw $t5, 4($t2)
    ori $s0, $zero, 2
    sw $t5, 4($t2)
```

```
occasion6: #ld_W_RT
```



```
lw $t6, 4($t0)
ori $s0, $zero, 1
ori $s1, $zero, 2
sw $t6, 0($t1)
```

```
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 0000000c
$11 <= 00000010
$12 <= 00000014
*00000000 <= 00000004
*00000004 <= 00000008
*00000008 <= 0000000c
*0000000c <= 00000010
$13 <= 00000008
*00000008 <= 0000000c
$14 <= 00000008
$16 <= 0000000c
*0000000c <= 00000010
$15 <= 0000000c
$17 <= 00000004
$18 <= 00000008
*0000000c <= 00000014
$13 <= 00000014
*00000004 <= 00000014
$13 <= 00000000
$16 <= 00000002
*00000010 <= 00000000
$14 <= 0000000c
$16 <= 00000001
$17 <= 00000002
*00000008 <= 0000000c
```

(jal\_sw 的情况)

## Beq:

**R\_E/M/W\_RS/RT(beq)(equal)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
subu $t4, $t0, $t1
```

```
occasion1: #R_E_RS
    addu $t5, $t0, $t1
    beq $t5, $t2, change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change1:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion2: #R_M_RS
    addu $t6, $t0, $t1
    ori $s0, $zero, 1
    beq $t6, $t2, change2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change2:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion3: #R_W_RS
    addu $t7, $t0, $t1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    beq $t7, $t2, change3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change3:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion4: #R_E_RT
    subu $t5, $t1, $t2
    beq $t4, $t5, change4
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change4:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion5: #R_M_RT
    subu $t6, $t1, $t2
    ori $s0, $zero, 1
    beq $t4, $t6, change5
    ori $s0, $zero, 1
```

```

ori $s1, $zero, 2
change5:
    ori $s2, $zero, 3
    ori $s3, $zero, 4

occasion6:  #R_W_RT
    subu $t7, $t1, $t2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    beq $t4, $t7, change6
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change6:
        ori $s2, $zero, 3
        ori $s3, $zero, 4

$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 0000000c
$11 <= 00000010
$12 <= ffffffff
$13 <= 0000000c
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$14 <= 0000000c
$16 <= 00000001
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$15 <= 0000000c
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$13 <= ffffffff
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$14 <= ffffffff
$16 <= 00000001
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004

```

```
$15 <= ffffffff
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
```

### **R\_E/M/W\_RS/RT(beq)(unequal)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
subu $t4, $t0, $t1
```

```
occasion1: #R_E_RS
    addu $t5, $t0, $t1
    beq $t5, $t3, change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change1:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion2: #R_M_RS
    addu $t6, $t0, $t1
    ori $s0, $zero, 1
    beq $t6, $t3, change2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change2:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion3: #R_W_RS
    addu $t7, $t0, $t1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    beq $t7, $t3, change3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change3:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```

occasion4:  #R_E_RT
    subu $t5, $t1, $t2
    beq $t3, $t5, change4
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change4:
    ori $s2, $zero, 3
    ori $s3, $zero, 4

```

```

occasion5:  #R_M_RT
    subu $t6, $t1, $t2
    ori $s0, $zero, 1
    beq $t3, $t6, change5
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change5:
    ori $s2, $zero, 3
    ori $s3, $zero, 4

```

```

occasion6:  #R_W_RT
    subu $t7, $t1, $t2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    beq $t3, $t7, change6
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change6:
    ori $s2, $zero, 3
    ori $s3, $zero, 4

```

\$ 8 <= 00000004

\$ 9 <= 00000008

\$10 <= 0000000c

\$11 <= 00000010

\$12 <= ffffffff

\$13 <= 0000000c

\$16 <= 00000001

\$17 <= 00000002

\$18 <= 00000003

\$19 <= 00000004

\$14 <= 0000000c

\$16 <= 00000001

\$16 <= 00000001

\$17 <= 00000002

\$18 <= 00000003

```

$19 <= 00000004
$15 <= 0000000c
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004
$13 <= ffffffff
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004
$14 <= ffffffff
$16 <= 00000001
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004
$15 <= ffffffff
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004

```

### **I\_E/M/W\_RS/RT(beq)(equal)**

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
subu $t4, $t0, $t1

```

```

occasion1:  #I_E_RS
    ori $t5, $zero, 16
    beq $t5, $t3, change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change1:

```

```
ori $s2, $zero, 3
ori $s3, $zero, 4
```

```
occasion2: #I_M_RS
ori $t6, $zero, 16
ori $s0, $zero, 1
beq $t6, $t3, change2
ori $s0, $zero, 1
ori $s1, $zero, 2
change2:
ori $s2, $zero, 3
ori $s3, $zero, 4
```

```
occasion3: #I_W_RS
ori $t7, $zero, 16
ori $s0, $zero, 1
ori $s1, $zero, 2
beq $t7, $t3, change3
ori $s0, $zero, 1
ori $s1, $zero, 2
change3:
ori $s2, $zero, 3
ori $s3, $zero, 4
```

```
occasion4: #I_E_RT
ori $t5, $zero, 8
beq $t1, $t5, change4
ori $s0, $zero, 1
ori $s1, $zero, 2
change4:
ori $s2, $zero, 3
ori $s3, $zero, 4
```

```
occasion5: #I_M_RT
ori $t6, $zero, 8
ori $s0, $zero, 1
beq $t1, $t6, change5
ori $s0, $zero, 1
ori $s1, $zero, 2
change5:
ori $s2, $zero, 3
ori $s3, $zero, 4
```

```
occasion6: #I_W_RT
```

```

ori $t7, $zero, 8
ori $s0, $zero, 1
ori $s1, $zero, 2
beq $t1, $t7, change6
ori $s0, $zero, 1
ori $s1, $zero, 2
change6:
    ori $s2, $zero, 3
    ori $s3, $zero, 4

```

```

$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 0000000c
$11 <= 00000010
$12 <= ffffffff
$13 <= 00000010
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$14 <= 00000010
$16 <= 00000001
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$15 <= 00000010
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$13 <= 00000008
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$14 <= 00000008
$16 <= 00000001
$16 <= 00000001
$18 <= 00000003
$19 <= 00000004
$15 <= 00000008
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$18 <= 00000003

```



\$19 <= 00000004

### **I\_E/M/W\_RS/RT(beq)(unequal)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $t3, $zero, 16
subu $t4, $t0, $t1
```

```
occasion1:  #I_E_RS
    ori $t5, $zero, 16
    beq $t5, $t2, change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change1:
        ori $s2, $zero, 3
        ori $s3, $zero, 4
```

```
occasion2:  #I_M_RS
    ori $t6, $zero, 16
    ori $s0, $zero, 1
    beq $t6, $t2, change2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change2:
        ori $s2, $zero, 3
        ori $s3, $zero, 4
```

```
occasion3:  #I_W_RS
    ori $t7, $zero, 16
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    beq $t7, $t2, change3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change3:
        ori $s2, $zero, 3
        ori $s3, $zero, 4
```

```
occasion4:  #I_E_RT
    ori $t5, $zero, 8
    beq $t2, $t5, change4
    ori $s0, $zero, 1
    ori $s1, $zero, 2
```

```
change4:
    ori $s2, $zero, 3
    ori $s3, $zero, 4
```

```
occasion5: #I_M_RT
    ori $t6, $zero, 8
    ori $s0, $zero, 1
    beq $t2, $t6, change5
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change5:
        ori $s2, $zero, 3
        ori $s3, $zero, 4
```

```
occasion6: #I_W_RT
    ori $t7, $zero, 8
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    beq $t2, $t7, change6
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change6:
        ori $s2, $zero, 3
        ori $s3, $zero, 4
```

\$ 8 <= 00000004

\$ 9 <= 00000008

\$10 <= 0000000c

\$11 <= 00000010

\$12 <= ffffffff

\$13 <= 00000010

\$16 <= 00000001

\$17 <= 00000002

\$18 <= 00000003

\$19 <= 00000004

\$14 <= 00000010

\$16 <= 00000001

\$16 <= 00000001

\$17 <= 00000002

\$18 <= 00000003

\$19 <= 00000004

\$15 <= 00000010

\$16 <= 00000001

\$17 <= 00000002

\$16 <= 00000001

```

$17 <= 00000002
$18 <= 00000003
$19 <= 00000004
$13 <= 00000008
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004
$14 <= 00000008
$16 <= 00000001
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004
$15 <= 00000008
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$19 <= 00000004

```

#### **Ld\_E/M/W\_RS/RT(beq)(equal)**

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $s0, $zero, 1
sw $t0, 0($zero)
sw $t1, 4($zero)
sw $t2, 8($zero)
ori $s0, $zero, 1
ori $s1, $zero, 2

```

```

occasion1: #ld_E_RS
    lw $t3, 0($t0)
    beq $t3, $t1, change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change1:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

```

occasion2: #ld_M_RS
    lw $t4, 0($t0)

```

```
ori $s2, $zero, 2
beq $t4, $t1, change2
ori $s0, $zero, 1
ori $s1, $zero, 2
change2:
    ori $s2, $zero, 2
    ori $s3, $zero, 3
```

```
occasion3: #ld_W_RS
    lw $t5, 0($t0)
    ori $s2, $zero, 2
    ori $s3, $zero, 3
    beq $t5, $t1, change3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change3:
        ori $s2, $zero, 2
        ori $s3, $zero, 3
```

```
occasion4: #ld_E_RT
    lw $t6, 0($t0)
    beq $t6, $t1, change4
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change4:
        ori $s2, $zero, 2
        ori $s3, $zero, 3
```

```
occasion5: #ld_M_RT
    lw $t7, 0($t0)
    ori $s2, $zero, 2
    beq $t7, $t1, change5
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change5:
        ori $s2, $zero, 2
        ori $s3, $zero, 3
```

```
occasion6: #ld_W_RT
    lw $t8, 0($t0)
    ori $s2, $zero, 2
    ori $s3, $zero, 3
    beq $t8, $t1, change6
    ori $s0, $zero, 1
```

```
ori $s1, $zero, 2
change6:
    ori $s2, $zero, 2
    ori $s3, $zero, 3
```

```
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 0000000c
$16 <= 00000001
*00000000 <= 00000004
*00000004 <= 00000008
*00000008 <= 0000000c
$16 <= 00000001
$17 <= 00000002
$11 <= 00000008
$16 <= 00000001
$18 <= 00000002
$19 <= 00000003
$12 <= 00000008
$18 <= 00000002
$16 <= 00000001
$18 <= 00000002
$19 <= 00000003
$13 <= 00000008
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001
$18 <= 00000002
$19 <= 00000003
$14 <= 00000008
$16 <= 00000001
$18 <= 00000002
$19 <= 00000003
$15 <= 00000008
$18 <= 00000002
$16 <= 00000001
$18 <= 00000002
$19 <= 00000003
$24 <= 00000008
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001
$18 <= 00000002
$19 <= 00000003
```

### **Ld\_E/M/W\_RS/RT(beq)(unequal)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 12
ori $s0, $zero, 1
sw $t0, 0($zero)
sw $t1, 4($zero)
sw $t2, 8($zero)
ori $s0, $zero, 1
ori $s1, $zero, 2
```

```
occasion1: #ld_E_RS
    lw $t3, 0($t0)
    beq $t3, $t0, change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change1:
    ori $s2, $zero, 2
    ori $s3, $zero, 3
```

```
occasion2: #ld_M_RS
    lw $t4, 0($t0)
    ori $s2, $zero, 2
    beq $t4, $t0, change2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change2:
    ori $s2, $zero, 2
    ori $s3, $zero, 3
```

```
occasion3: #ld_W_RS
    lw $t5, 0($t0)
    ori $s2, $zero, 2
    ori $s3, $zero, 3
    beq $t5, $t0, change3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change3:
    ori $s2, $zero, 2
    ori $s3, $zero, 3
```

```
occasion4: #ld_E_RT
    lw $t6, 0($t0)
```

```

beq $t0, $t6, change4
ori $s0, $zero, 1
ori $s1, $zero, 2
change4:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

```

occasion5: #ld_M_RT
lw $t7, 0($t0)
ori $s2, $zero, 2
beq $t0, $t7, change5
ori $s0, $zero, 1
ori $s1, $zero, 2
change5:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

```

occasion6: #ld_W_RT
lw $t8, 0($t0)
ori $s2, $zero, 2
ori $s3, $zero, 3
beq $t0, $t8, change6
ori $s0, $zero, 1
ori $s1, $zero, 2
change6:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

```

$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 0000000c
$16 <= 00000001
*00000000 <= 00000004
*00000004 <= 00000008
*00000008 <= 0000000c
$16 <= 00000001
$17 <= 00000002
$11 <= 00000008
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$12 <= 00000008
$18 <= 00000002
$16 <= 00000001

```

```

$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$13 <= 00000008
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$14 <= 00000008
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$15 <= 00000008
$18 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$24 <= 00000008
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003

```

### **Jal\_M/W\_RS/RT(beq)(equal)**

```

ori $t0, $zero, 4
ori $t1, $zero, 0x00003014
ori $t2, $zero, 0x00003034
ori $t3, $zero, 0x00003058
ori $t4, $zero, 0x00003078
occasion1: #jal_M_RS
    jal change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change1:
    beq $ra, $t1, change11
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change11:

```



```

        ori $s2, $zero, 2
        ori $s3, $zero, 3
occasion2: #jal_W_RS
jal change2
ori $s0, $zero, 1
ori $s1, $zero, 2
change2:
    ori $s2, $zero, 2
    beq $ra, $t2, change21
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change21:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

```

occasion3: #jal_M_RT
jal change3
ori $s0, $zero, 1
ori $s1, $zero, 2
change3:
    beq $t3, $ra, change31
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change31:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

```

occasion4: #jal_W_RT
jal change4
ori $s0, $zero, 1
ori $s1, $zero, 2
change4:
    ori $s2, $zero, 2
    beq $t4, $ra, change41
    ori $s0, $zero, 1
    ori $s1, $zero, 2
change41:
    ori $s2, $zero, 2
    ori $s3, $zero, 3

```

\$ 8 <= 00000004

\$ 9 <= 00003014

\$10 <= 00003034

\$11 <= 00003058

\$12 <= 00003078

\$31 <= 0000301c

```

$16 <= 00000001
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$31 <= 0000303c
$16 <= 00000001
$18 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$31 <= 00003060
$16 <= 00000001
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$31 <= 00003080
$16 <= 00000001
$18 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003

```

### **Jal\_M/W\_RS/RT(beq)(unequal)**

```

ori $t0, $zero, 4
ori $t1, $zero, 0x00003010
ori $t2, $zero, 0x00003030
ori $t3, $zero, 0x00003050
ori $t4, $zero, 0x00003070
occasion1: #jal_M_RS
    jal change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change1:
        beq $ra, $t1, change11
        ori $s0, $zero, 1
        ori $s1, $zero, 2
        change11:
            ori $s2, $zero, 2
            ori $s3, $zero, 3
occasion2: #jal_W_RS

```

```

jal change2
ori $s0, $zero, 1
ori $s1, $zero, 2
change2:
    ori $s2, $zero, 2
    beq $ra, $t2, change21
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change21:
        ori $s2, $zero, 2
        ori $s3, $zero, 3

occasion3: #jal_M_RT
jal change3
ori $s0, $zero, 1
ori $s1, $zero, 2
change3:
    beq $t3, $ra, change31
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change31:
        ori $s2, $zero, 2
        ori $s3, $zero, 3

occasion4: #jal_W_RT
jal change4
ori $s0, $zero, 1
ori $s1, $zero, 2
change4:
    ori $s2, $zero, 2
    beq $t4, $ra, change41
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    change41:
        ori $s2, $zero, 2
        ori $s3, $zero, 3

```

\$ 8 <= 00000004

\$ 9 <= 00003010

\$10 <= 00003030

\$11 <= 00003050

\$12 <= 00003070

\$31 <= 0000301c

\$16 <= 00000001

\$16 <= 00000001

```

$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$31 <= 0000303c
$16 <= 00000001
$18 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$31 <= 00003060
$16 <= 00000001
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$31 <= 00003080
$16 <= 00000001
$18 <= 00000002
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003

```

### **Jal\_M\_RS(jr)**

```

ori $t0, $zero, 4
ori $t1, $zero, 0x00003010
ori $t2, $zero, 0x00003030
ori $t3, $zero, 0x00003050
ori $t4, $zero, 0x00003070
occasion1: #jal_M_RS
    jal change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    ori $s2, $zero, 2
    ori $s3, $zero, 3
    change1:
        jr $ra
        ori $s0, $zero, 1
        ori $s1, $zero, 2
$ 8 <= 00000004
$ 9 <= 00003010
$10 <= 00003030

```

```

$11 <= 00003050
$12 <= 00003070
$31 <= 0000301c
$16 <= 00000001
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001
$17 <= 00000002
$18 <= 00000002
$19 <= 00000003
$16 <= 00000001

```

### **Jal\_W\_RS(jr)**

```

ori $t0, $zero, 4
ori $t1, $zero, 0x00003010
ori $t2, $zero, 0x00003030
ori $t3, $zero, 0x00003050
ori $t4, $zero, 0x00003070
occasion1: #jal_W_RS
    jal change1
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    ori $s2, $zero, 2
    ori $s3, $zero, 3
    change1:
        ori $s3, $zero, 3
        jr $ra
        ori $s0, $zero, 1
        ori $s1, $zero, 2

```

```

$ 8 <= 00000004
$ 9 <= 00003010
$10 <= 00003030
$11 <= 00003050
$12 <= 00003070
$31 <= 0000301c
$16 <= 00000001
$19 <= 00000003
$16 <= 00000001

```

\$17 <= 00000002  
\$18 <= 00000002  
\$19 <= 00000003  
\$19 <= 00000003  
\$16 <= 00000001  
\$17 <= 00000002  
\$18 <= 00000002  
\$19 <= 00000003  
\$19 <= 00000003

ori \$t0, \$zero, 4  
ori \$t1, \$zero, 8  
ori \$t2, \$zero, 0x00003000  
ori \$s0, \$zero, 1  
ori \$s1, \$zero, 2  
ori \$s2, \$zero, 3  
occasion1: #R\_E\_RS  
    addu \$t3, \$t0, \$t2  
    jr \$t3  
    ori \$s0, \$zero, 1  
    ori \$s1, \$zero, 2  
    ori \$s2, \$zero, 3

\$ 8 <= 00000004  
\$ 9 <= 00000008  
\$10 <= 00003000  
\$16 <= 00000001  
\$17 <= 00000002  
\$18 <= 00000003  
\$11 <= 00003004  
\$16 <= 00000001  
\$ 9 <= 00000008  
\$10 <= 00003000  
\$16 <= 00000001  
\$17 <= 00000002  
\$18 <= 00000003  
\$11 <= 00003004  
\$16 <= 00000001  
\$ 9 <= 00000008  
\$10 <= 00003000

### **R\_M\_RS(jr)**

ori \$t0, \$zero, 4  
ori \$t1, \$zero, 8

```

ori $t2, $zero, 0x00003000
ori $s0, $zero, 1
ori $s1, $zero, 2
ori $s2, $zero, 3
occasion1:  #R_M_RS
    addu $t3, $t0, $t2
    ori $s0, $zero, 1
    jr $t3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    ori $s2, $zero, 3
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 00003000
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$11 <= 00003004
$16 <= 00000001
$16 <= 00000001
$ 9 <= 00000008
$10 <= 00003000
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$11 <= 00003004
$16 <= 00000001
$16 <= 00000001
$ 9 <= 00000008
$10 <= 00003000

```

### **R\_W\_RS(jr)**

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 0x00003000
ori $s0, $zero, 1
ori $s1, $zero, 2
ori $s2, $zero, 3
occasion1:  #R_W_RS
    addu $t3, $t0, $t2
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    jr $t3
    ori $s0, $zero, 1

```

```

        ori $s1, $zero, 2
        ori $s2, $zero, 3
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 00003000
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$11 <= 00003004
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001
$ 9 <= 00000008
$10 <= 00003000
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$11 <= 00003004
$16 <= 00000001
$17 <= 00000002
$16 <= 00000001

```

### **I\_E\_RS(jr)**

```

ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 0x00003000
ori $s0, $zero, 1
ori $s1, $zero, 2
ori $s2, $zero, 3
occasion1: #R_E_RS
        ori $t3, $t2, 0
        jr $t3
        ori $s0, $zero, 1
        ori $s1, $zero, 2
        ori $s2, $zero, 3
$ 8 <= 00000004
$ 9 <= 00000008
$10 <= 00003000
$16 <= 00000001
$17 <= 00000002
$18 <= 00000003
$11 <= 00003000
$16 <= 00000001
$ 8 <= 00000004

```



\$ 9 <= 00000008  
\$10 <= 00003000  
\$16 <= 00000001  
\$17 <= 00000002  
\$18 <= 00000003  
\$11 <= 00003000  
\$16 <= 00000001  
\$ 8 <= 00000004

### **I\_M\_RS(jr)**

ori \$t0, \$zero, 4  
ori \$t1, \$zero, 8  
ori \$t2, \$zero, 0x00003000  
ori \$s0, \$zero, 1  
ori \$s1, \$zero, 2  
ori \$s2, \$zero, 3  
occasion1: #R\_M\_RS  
    ori \$t3, \$t2, 0  
    ori \$s0, \$zero, 1  
    jr \$t3  
    ori \$s0, \$zero, 1  
    ori \$s1, \$zero, 2  
    ori \$s2, \$zero, 3

\$ 8 <= 00000004  
\$ 9 <= 00000008  
\$10 <= 00003000  
\$16 <= 00000001  
\$17 <= 00000002  
\$18 <= 00000003  
\$11 <= 00003000  
\$16 <= 00000001  
\$16 <= 00000001  
\$ 8 <= 00000004  
\$ 9 <= 00000008  
\$10 <= 00003000  
\$16 <= 00000001  
\$17 <= 00000002  
\$18 <= 00000003  
\$11 <= 00003000  
\$16 <= 00000001  
\$16 <= 00000001  
\$ 8 <= 00000004

### **I\_W\_RS(jr)**

```
ori $t0, $zero, 4
ori $t1, $zero, 8
ori $t2, $zero, 0x00003000
ori $s0, $zero, 1
ori $s1, $zero, 2
ori $s2, $zero, 3
occasion1: #R_W_RS
    ori $t3, $t2, 0
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    jr $t3
    ori $s0, $zero, 1
    ori $s1, $zero, 2
    ori $s2, $zero, 3
$ 8 <= 00000004
$ 9 <= 00000008
```

乘除测试代码:

```
lui $t0, 0x7000
lui $t1, 0xc000
ori $t0, $t0, 0x1234
ori $t1, $t1, 0x1234
mult $t0, $t1
mfhi $t2
mflo $t3
multu $t0, $t1
mfhi $t2
mflo $t3
div $t1, $t0
mfhi $t2
mflo $t3
divu $t1, $t0
mfhi $t2
mflo $t3
```

mult \$t0, \$t1

mtlo \$t2

mthi \$t3

mfhi \$t2

mflo \$t3

预期输出：

45@00003000: \$ 8 <= 70000000

55@00003004: \$ 9 <= c0000000

65@00003008: \$ 8 <= 70001234

75@0000300c: \$ 9 <= c0001234

145@00003014: \$10 <= e4000369

155@00003018: \$11 <= c14b5a90

225@00003020: \$10 <= 5400159d

235@00003024: \$11 <= c14b5a90

355@0000302c: \$10 <= c0001234

365@00003030: \$11 <= 00000000

485@00003038: \$10 <= 50000000

495@0000303c: \$11 <= 00000001

585@0000304c: \$10 <= 00000001

595@00003050: \$11 <= 50000000

测试程序：

# R test

li \$1, 100

li \$2, 60

add \$3, \$1, \$2

sw \$3, -4(\$3)

sw \$3, 0(\$0)

li \$3, 12

sb \$3, -2(\$3)

li \$1, -1

li \$2, -3

add \$3, \$2, \$1

li \$2, 0x80000000

lui \$1, 0x8000

addu \$3, \$1, \$2

sw \$1, 4(\$3)

sh \$2, 2(\$0)

#add \$3, \$1, \$2

lui \$1, 0x7700

lui \$2, 0x7001

addu \$3, \$1, \$2

#add \$3, \$1, \$2

li \$1, 100

li \$2, 150

sub \$3, \$1, \$2

addu \$3, \$3, \$1

sh \$3, 2(\$3)

lui \$1, 0x7fff

lui \$1, 0x8000

subu \$3, \$1, \$2

#sub \$3, \$1, \$2

li \$1, 100

li \$2, 50

mult \$1, \$2

mfhi \$3

mflo \$4  
sb \$2, -19(\$4)  
sb \$2, 7(\$3)

lui \$1, 0x8000  
lui \$2, 100  
mult \$1, \$2  
mfhi \$3  
mflo \$4

lui \$1, 0x8000  
lui \$2, 0x8000  
mult \$1, \$2  
mfhi \$3  
mflo \$4

lui \$1, 100  
lui \$2, 0x8000  
mult \$1, \$2  
mfhi \$3  
mflo \$4

li \$1, 100  
li \$2, 50  
multu \$1, \$2  
mfhi \$3  
mflo \$4

lui \$1, 0x8000  
lui \$2, 100

multu \$1, \$2  
mfhi \$3  
mflo \$4  
sw \$3, 0(\$0)  
sb \$3, 100(\$0)

lui \$1, 0x8000  
lui \$2, 0x8000  
multu \$1, \$2  
mfhi \$3  
li \$5, 200  
sw \$3, 0(\$5)  
mflo \$4

lui \$1, 100  
lui \$2, 0x8000  
multu \$1, \$2  
mfhi \$3  
mflo \$4

li \$1, -8  
li \$2, -4  
div \$1, \$2  
mfhi \$3  
li \$2, 64  
sw \$3, -16(\$2)  
mflo \$4

li \$1, 16  
li \$2, -6

div \$1, \$2  
mfhi \$3  
li \$3, 100  
mflo \$4  
sh \$4, 2(\$3)

li \$1, 16  
li \$2, 5  
div \$1, \$2  
mfhi \$3  
li \$5, 100  
mflo \$4  
sh \$4, 8(\$5)

li \$1, -16  
li \$2, 4  
div \$1, \$2  
mfhi \$3  
mflo \$4

li \$1, -8  
li \$2, -4  
divu \$1, \$2  
mfhi \$3  
mflo \$4

li \$1, 16  
li \$2, -6  
divu \$1, \$2  
mfhi \$3

li \$5, 100  
mflo \$4  
sh \$4, 8(\$5)

li \$1, 16  
li \$2, 5  
divu \$1, \$2  
mfhi \$3  
mflo \$4

li \$1, -16  
li \$2, 4  
divu \$1, \$2  
mfhi \$3  
mflo \$4

li \$1, 80  
li \$2, 70  
slt \$3, \$1, \$2  
slt \$4, \$2, \$1

li \$1, -10  
li \$2, 100  
slt \$3, \$1, \$2  
slt \$3, \$2, \$1

li \$1, -100  
li \$2, -50  
slt \$3, \$1, \$2  
slt \$3, \$2, \$1



```
li $1, -20
li $2, 100
sltu $3, $1, $2      # -20<100
sltu $3, $2, $1      #
sltu $3, $1, $0
sltu $3, $0, $1
```

```
li $1, 0
sltu $3, $1, $1
sltu $3, $1, $0
li $1, 100
sltu $3, $1, $0
```

```
li $9, 128
sll $10, $9, 3
lui $12, 0xff00
srl $13, $12, 2
lui $5, 0xff00
sra $6, $5, 2
li $8, 12
li $5, 4
srav $6, $8, $5
li $9, 13
srlv $1, $9, $8
li $10, 6
li $5, 3
sllv $3, $10, $5
```

```
li $1, 0x1008
```

```
li $2, 0xdff7
xor $3, $1, $2
```

```
xori $3, $0, 0x8fff
li $1, 0xffff
li $2, 0x1000
or $2, $1, $2
```

```
li $1, 100
li $2, 50
sw $2, 0($1)
```

```
li $1, 50
li $2, 100
sw $1, 4($2)
```

预期结果:

```
45@00003000: $ 1 <= 00000064
55@00003004: $ 2 <= 0000003c
65@00003008: $ 3 <= 000000a0
65@0000300c: *0000009c <= 000000a0
75@00003010: *00000000 <= 000000a0
95@00003014: $ 3 <= 0000000c
95@00003018: *00000008 <= 000c0000
115@0000301c: $ 1 <= ffffffff
125@00003020: $ 2 <= ffffffff
135@00003024: $ 3 <= ffffffff
145@00003028: $ 1 <= 80000000
155@0000302c: $ 2 <= 80000000
165@00003030: $ 1 <= 80000000
175@00003034: $ 3 <= 00000000
175@00003038: *00000004 <= 80000000
185@0000303c: *00000000 <= 000000a0
205@00003040: $ 1 <= 77000000
215@00003044: $ 2 <= 70010000
225@00003048: $ 3 <= e7010000
```

235@0000304c: \$ 1 <= 00000064  
245@00003050: \$ 2 <= 00000096  
255@00003054: \$ 3 <= ffffffce  
265@00003058: \$ 3 <= 00000032  
265@0000305c: \*00000034 <= 00000032  
285@00003060: \$ 1 <= 7fff0000  
295@00003064: \$ 1 <= 80000000  
305@00003068: \$ 3 <= 7ffff6a  
315@0000306c: \$ 1 <= 00000064  
325@00003070: \$ 2 <= 00000032  
395@00003078: \$ 3 <= 00000000  
405@0000307c: \$ 4 <= 00001388  
405@00003080: \*00001374 <= 00003200  
415@00003084: \*00000004 <= 32000000  
435@00003088: \$ 1 <= 80000000  
445@0000308c: \$ 2 <= 00640000  
515@00003094: \$ 3 <= ffce0000  
525@00003098: \$ 4 <= 00000000  
535@0000309c: \$ 1 <= 80000000  
545@000030a0: \$ 2 <= 80000000  
615@000030a8: \$ 3 <= 40000000  
625@000030ac: \$ 4 <= 00000000  
635@000030b0: \$ 1 <= 00640000  
645@000030b4: \$ 2 <= 80000000  
715@000030bc: \$ 3 <= ffce0000  
725@000030c0: \$ 4 <= 00000000  
735@000030c4: \$ 1 <= 00000064  
745@000030c8: \$ 2 <= 00000032  
815@000030d0: \$ 3 <= 00000000  
825@000030d4: \$ 4 <= 00001388  
835@000030d8: \$ 1 <= 80000000  
845@000030dc: \$ 2 <= 00640000  
915@000030e4: \$ 3 <= 00320000  
925@000030e8: \$ 4 <= 00000000  
925@000030ec: \*00000000 <= 00320000  
935@000030f0: \*00000064 <= 00000000  
955@000030f4: \$ 1 <= 80000000  
965@000030f8: \$ 2 <= 80000000  
1035@00003100: \$ 3 <= 40000000  
1045@00003104: \$ 5 <= 000000c8  
1045@00003108: \*000000c8 <= 40000000  
1065@0000310c: \$ 4 <= 00000000  
1075@00003110: \$ 1 <= 00640000  
1085@00003114: \$ 2 <= 80000000

1155@0000311c: \$ 3 <= 00320000  
1165@00003120: \$ 4 <= 00000000  
1175@00003124: \$ 1 <= ffffffff8  
1185@00003128: \$ 2 <= ffffffff c  
1305@00003130: \$ 3 <= 00000000  
1315@00003134: \$ 2 <= 00000040  
1315@00003138: \*00000030 <= 00000000  
1335@0000313c: \$ 4 <= 00000002  
1345@00003140: \$ 1 <= 00000010  
1355@00003144: \$ 2 <= ffffffff a  
1475@0000314c: \$ 3 <= 00000004  
1485@00003150: \$ 3 <= 00000064  
1495@00003154: \$ 4 <= ffffffff e  
1495@00003158: \*00000064 <= fffe0000  
1515@0000315c: \$ 1 <= 00000010  
1525@00003160: \$ 2 <= 00000005  
1645@00003168: \$ 3 <= 00000001  
1655@0000316c: \$ 5 <= 00000064  
1665@00003170: \$ 4 <= 00000003  
1665@00003174: \*0000006c <= 00000003  
1685@00003178: \$ 1 <= ffffffff 0  
1695@0000317c: \$ 2 <= 00000004  
1815@00003184: \$ 3 <= 00000000  
1825@00003188: \$ 4 <= ffffffff c  
1835@0000318c: \$ 1 <= ffffffff 8  
1845@00003190: \$ 2 <= ffffffff c  
1965@00003198: \$ 3 <= ffffffff 8  
1975@0000319c: \$ 4 <= 00000000  
1985@000031a0: \$ 1 <= 00000010  
1995@000031a4: \$ 2 <= ffffffff a  
2115@000031ac: \$ 3 <= 00000010  
2125@000031b0: \$ 5 <= 00000064  
2135@000031b4: \$ 4 <= 00000000  
2135@000031b8: \*0000006c <= 00000000  
2155@000031bc: \$ 1 <= 00000010  
2165@000031c0: \$ 2 <= 00000005  
2285@000031c8: \$ 3 <= 00000001  
2295@000031cc: \$ 4 <= 00000003  
2305@000031d0: \$ 1 <= ffffffff 0  
2315@000031d4: \$ 2 <= 00000004  
2435@000031dc: \$ 3 <= 00000000  
2445@000031e0: \$ 4 <= 3fffffff c  
2455@000031e4: \$ 1 <= 00000050  
2465@000031e8: \$ 2 <= 00000046

2475@000031ec: \$ 3 <= 00000000  
2485@000031f0: \$ 4 <= 00000001  
2495@000031f4: \$ 1 <= fffffff6  
2505@000031f8: \$ 2 <= 00000064  
2515@000031fc: \$ 3 <= 00000001  
2525@00003200: \$ 3 <= 00000000  
2535@00003204: \$ 1 <= ffffff9c  
2545@00003208: \$ 2 <= ffffffce  
2555@0000320c: \$ 3 <= 00000001  
2565@00003210: \$ 3 <= 00000000  
2575@00003214: \$ 1 <= ffffffec  
2585@00003218: \$ 2 <= 00000064  
2595@0000321c: \$ 3 <= 00000000  
2605@00003220: \$ 3 <= 00000001  
2615@00003224: \$ 3 <= 00000000  
2625@00003228: \$ 3 <= 00000001  
2635@0000322c: \$ 1 <= 00000000  
2645@00003230: \$ 3 <= 00000000  
2655@00003234: \$ 3 <= 00000000  
2665@00003238: \$ 1 <= 00000064  
2675@0000323c: \$ 3 <= 00000000  
2685@00003240: \$ 9 <= 00000080  
2695@00003244: \$10 <= 00000400  
2705@00003248: \$12 <= ff000000  
2715@0000324c: \$13 <= 3fc00000  
2725@00003250: \$ 5 <= ff000000  
2735@00003254: \$ 6 <=ffc00000  
2745@00003258: \$ 8 <= 0000000c  
2755@0000325c: \$ 5 <= 00000004  
2765@00003260: \$ 6 <= 00000000  
2775@00003264: \$ 9 <= 0000000d  
2785@00003268: \$ 1 <= 00000000  
2795@0000326c: \$10 <= 00000006  
2805@00003270: \$ 5 <= 00000003  
2815@00003274: \$ 3 <= 00000030  
2825@00003278: \$ 1 <= 00001008  
2835@0000327c: \$ 2 <= 0000dff7  
2845@00003280: \$ 3 <= 0000cfff  
2855@00003284: \$ 3 <= 00008fff  
2865@00003288: \$ 1 <= 0000ffff  
2875@0000328c: \$ 2 <= 00001000  
2885@00003290: \$ 2 <= 0000ffff  
2895@00003294: \$ 1 <= 00000064  
2905@00003298: \$ 2 <= 00000032

2905@0000329c: \*00000064 <= 00000032  
2925@000032a0: \$ 1 <= 00000032  
2935@000032a4: \$ 2 <= 00000064  
2935@000032a8: \*00000068 <= 00000032

测试程序:

```
ori  $t0,  $0,  0x1234
mthi  $t0
mtlo  $t0
mfhi  $t1
mflo  $t2
```

预期结果:

45@00003000: \$ 8 <= 00001234  
75@0000300c: \$ 9 <= 00001234  
85@00003010: \$10 <= 00001234

测试程序:

.text

```
ori $t1, $0, 3
jal lable1
nop
add $t1, $t1, $t1
ori $t2, $0, 0x3024
jr $t2
```

lable1:

```
ori $t0, $0, 2
jalr $a0, $31
nop

nop
```

预期结果:

```
45@00003000: $ 9 <= 00000003
55@00003004: $31 <= 0000300c
75@00003018: $ 8 <= 00000002
85@0000301c: $31 <= 00003024
105@0000300c: $ 9 <= 00000006
115@00003010: $10 <= 00003024
145@00003018: $ 8 <= 00000002
```

测试程序:

```
ori $t1, $0, 3
jal lable1
nop
add $t1, $t1, $t1
ori $t2, $0, 0x3024
jr $t2
```

lable1:

```
ori $t0, $0, 2
jalr $31, $31
nop
```

Nop

预期结果:

```
45@00003000: $ 9 <= 00000003
55@00003004: $31 <= 0000300c
75@00003018: $ 8 <= 00000002
85@0000301c: $31 <= 00003024
105@0000300c: $ 9 <= 00000006
115@00003010: $10 <= 00003024
145@00003018: $ 8 <= 00000002
```